The DigiHome Service-Oriented Platform

Daniel Romero, Gabriel Hermosillo, Amirhosein Taherkordi, Russel Nzekwa, Romain Rouvoy, Frank Eliassen

SOFTWARE—PRACTICE AND EXPERIENCE
Softw. Pract. Exper. 2010; **00**:1–27 *Prepared using* speauth.cls *[Version: 2002/09/23 v2.2]*

The DigiHome Service-Oriented Platform

Daniel Romero[1], Gabriel Hermosillo[1], Amirhosein Taherkordi[2], Russel Nzekwa[1], Romain Rouvoy[1], Frank Eliassen[2]

[1] *ADAM Project-team, University Lille 1, LIFL UMR CNRS 8022, INRIA Lille – Nord Europe, F-59650 Villeneuve d'Ascq*
[2] *Department of Informatics, University of Oslo, N-0316 Oslo*

SUMMARY

Nowadays, the computational devices are everywhere. In malls, offices, streets, cars and even in homes we can find devices providing and consuming functionality in order to improve the user satisfaction. These devices include sensors that provide information about the environment state (*e.g.*, temperature, occupancy, light levels), service providers (*e.g.*, Internet TVs, GPS), smartphones (that contain user preferences), and actuators that act on the environment (*e.g.*, closing the blinds, activating the alarm, changing the temperature). Although these devices exhibit communication capabilities, their integration into a larger monitoring system remains a challenging task, partly due to the strong heterogeneity of technologies and protocols. Therefore, in this article we focus on home environments and propose a middleware solution, called DigiHome, which applies the SCA (*Service Component Architecture*) component model in order to integrate data and events generated by heterogeneous devices in this kind of environments. DigiHome exploits the SCA extensibility to incorporate the REST (*REpresentational State Transfer*) architectural style, and in this way leverages on the integration of multi-scale systems-of-systems (from Wireless Sensor Networks to the Internet). Additionally, the platform applies CEP (*Complex Event Processing*) technology that detects application-specific situations. We claim that the modularization of concerns fostered by DigiHome and materialized in a service-oriented architecture, makes it easier to incorporate new services and devices in smart home environments. The benefits of the DigiHome platform are demonstrated on smart home scenarios covering home automation, emergency detection, and energy saving situations.

1. Introduction

Pervasive environments support context-aware applications that adapt their behavior by reasoning dynamically about the user and the surrounding information. This contextual information generally comes from diverse and heterogeneous entities, such as physical devices, *Wireless Sensors Networks* (WSNs), and smartphones. In order to exploit the information provided by these entities, a middleware solution is required to collect, process, and distribute the contextual information efficiently. However, the heterogeneity of systems in terms of

Received received-date
Revised revised-date

technology capabilities and communication protocols, the mobility of the different interacting entities, and the identification of adaptation situations make this integration difficult. Thus, this challenge requires a flexible solution in terms of communication support and context processing to leverage context-aware applications on the integration of heterogeneous context providers.

In particular, a solution dealing with context information and control environments must be able to connect with a wide range of device types. However, the resource scarcity in WSNs and mobile devices makes the development of such a solution very challenging. In this article, we propose the DigiHome platform, an improved version of our work introduced in [1]. With this platform we provide a simple but efficient service-oriented middleware solution to facilitate context-awareness in pervasive environments. Specifically, DigiHome supports the *integration*, *processing* and *adaptation* of the context-aware applications. Our solution enables the integration of heterogeneous computational entities by relying on the Service Component Architecture (SCA) model [2], the REST (*REpresentational State Transfer*) principles [3], standard discovery and communication protocols, and resource representation formats. We combined SCA and REST in our solution in order to foster reuse and loose coupling between the different services that compose the platform. Furthermore, while our solution also benefits from WSNs to operate simple event reasoning on the sensor nodes, we rely on *Complex Event Processing* [4] for analyzing in real-time the relationships between the different collected events and trigger rule-based adaptations.

The remainder of this article is organized as follows. We start by describing a smart home scenario in which we identify the key challenges in pervasive environments that motivate this work (cf. section 2). Then, we present some of the background concepts that we use in our project (cf. section 3). We continue by the description of DigiHome, our middleware platform to support the integration of systems-of-systems in pervasive environments (cf. section 4). Then, we discuss the benefits of our approach (cf. section 5) before presenting the related work (cf. section 6). Finally, we conclude by presenting some promising perspectives for this work (cf. section 7).

2. Motivating Scenario

In this paper, we use a smart home scenario to show the motivation of our work. A smart home generally refers to a house environment equipped with several types of computing entities, such as *sensors*, which collect physical information (temperature, movement detection, noise level, light, etc.), and *actuators*, which change the state of the environment. Sensor nodes are mostly embedded in home appliances and may be powered by batteries with limited capacity. In this scenario, we consider a smart home equipped with occupancy, smoke detection, and temperature sensors. These tiny devices have the ability to collect context information and to communicate wirelessly with each other, in order to identify the context situation of the environment. In addition to that, we can also use actuators to physically control lights, TV, and air conditioning. Figure 1 illustrates the integration of these sensors and actuators in our scenario. As observed in this figure, the different entities use heterogeneous protocols to interact. In the scenario, the smartphones provide information about the user preferences for

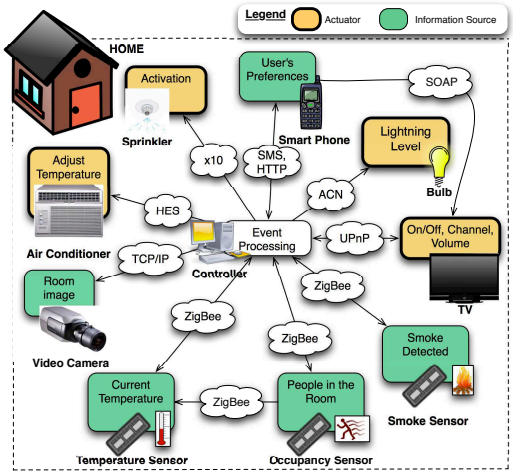

Figure 1. Interactions between the smart home devices.

the home configuration. When several people share the same room, the configuration decision is based on merged preferences. Conflicts between the user preferences are resolved by giving, *e.g.*, priority to the person who arrived first to the room. The mobile devices also have an application that enables the control of the actuators present in the different rooms. This application can be adapted when there are changes in the actuator's configuration. Finally, there is a *Controller* device, which is able to gather information, and interact with the other co-located devices.

To show how the different elements of our scenario interact, we present three different situations:

Situation 1: Alice arrives to the living room. The occupancy sensor detects her presence and triggers the temperature sensors to increase the sampling rate of data. It also notifies the Controller that the room is occupied by somebody, which in turn tries to identify the occupant by looking for a profile in her mobile device. When Alice's profile is found, the Controller loads it and adjusts the temperature and lightening level of the room according to Alice's preferences.

Situation 2: The sensors detect smoke and notify the Controller, which using the occupancy sensor, detects that the house is empty. The Controller therefore sends an SMS to Alice, including a picture of the room captured using the surveillance camera. After checking the picture, Alice decides to remotely trigger the sprinklers using her mobile device. She also tells the system to alert the fire department about the problem. If Alice does not reply to the Controller within 5 minutes, the system activates automatically the sprinklers and alerts the fire department.

Situation 3: Alice installs a new TV in the bedroom. The Controller detects the presence of the new device, identifies it, and downloads the corresponding control software from an Internet repository. The platform tries to locate the available mobile devices, using a discovery protocol, and finds Alice's mobile device. The Controller proposes to update the mobile device with the components for controlling the new TV.

2.1. Key Challenges.

The various situations we described above allow us to identify several key challenges in terms of:

1. *Integration of multi-scale entities*: The mobile devices and sensors have different hardware and software capabilities, which make some devices more powerful than others. Therefore, the integration of these entities requires a flexible and simple solution that supports multiple interaction mechanisms and considers the restricted capabilities of some devices. In particular, regarding sensor nodes, the immaturity of high-level communication protocols, as well as the inherent resource scarcity, bring two critical challenges to our work: *1)* how sensor nodes should be connected to mobile devices and actuators through a standard high-level communication protocol, and *2)* the framework which runs over sensor nodes for supporting context-awareness and adaptation should not impose high resource demands.
2. *Entities mobility*: In our scenario, computational entities appear and disappear constantly. In particular, mobile devices providing user profiles are not always accessible (they can be turned off or the owner can leave the house with them). In a similar way, the actuators can be replaced or new ones can be added. Thus, we need to discover new entities dynamically as well as to support device disconnections.
3. *Information processing and adaptation*: In order to support adaptation, we first need to identify the situations, in which the adaptation is required. We have a lot of information that is generated by the different devices in the environment. Therefore, we need to define which part of this information is useful to identify relevant situations and react accordingly. In our scenario, those situations include the load of Alice's profile and the adjustment of the temperature, the sending of alerts via SMS in case of an emergency, and the adaptation of Alice's mobile device to control the new TV in her bedroom.

3. Background

In this section we present a brief introduction to the main elements employed in the DigiHome conception: The Service Component Architecture Model (SCA), the FraSCAti platform and Complex Event Processing technology.

3.1. Service Component Architecture (SCA) Model

SCA is a set of specifications for building distributed applications based on *Service-oriented architecture* (SOA) and *Component-Based Software Engineering* (CBSE) principles [2]. In SCA, the basic construction blocks are the *software components*, which have *services* (or provided interfaces), *references* (or required interfaces) and exposed properties. The references and services are connected by means of *wires*. SCA specifies a hierarchical component model, which means that components can be implemented either by primitive language entities or by subcomponents. In the latter case the components are called *composites*.

SCA is designed to be independent from programming languages, *Interface Definition Languages* (IDL), communication protocols and non-functional properties. In this way, an SCA-based application can be built, for example, using components in Java, PHP, and COBOL. Furthermore, several IDLs are supported, such as WSDL and Java Interfaces. In order to support interaction via different communication protocols, SCA provides the concept of *binding*. For SCA references, *bindings* describe the access mechanism used to call a service. In the case of services, the bindings describe the access mechanism that clients have to use to call the service.

3.2. The FRASCATI platform

The FraSCAti platform [5, 6] allows the development and execution of SCA-based applications. The platform itself is built as an SCA application—*i.e.*, its different subsystems are implemented as SCA components. FraSCAti provides an homogeneous view of a middleware software stack where the platform, the non-functional services, and the applications are uniformly designed and implemented with the same component-based and service-oriented paradigm. To achieve this, the platform is composed of four layers: *i)* the *Kernel Level* based on Fractal [7], a lightweight and open component framework with basic dependency injection, introspection and reconfiguration capabilities; *ii)* the *Personality Level*, which customizes the component kernel by providing the components with execution semantics and implementing the SCA API and principles based on the Fractal component model; *iii)* the *Runtime Level* that instantiates SCA assemblies and components and defines a flexible configuration process, which is inspired by the *extender* and *whiteboard* [8] design patterns of OSGi; and *iv)* the *Non-Functional Level* that supports the SCA Policy Framework specification in order to provide non-functional services implemented as regular SCA components.

3.3. Complex Event Processing

CEP is an emerging technology for finding relationships between series of simple and independent events from different sources, using previously defined rules [4]. It employs different techniques such as detection of complex patterns, event correlation and abstraction, event hierarchies and relationships between events using causality, membership, and timing. It is used in a variety of domains such as logistics, transport and finance. In our scenario, we consider a lot of heterogeneous devices (sensors, mobile devices, etc.) that generate isolated events, which can be used to obtain valuable information and to make decisions accordingly.

To understand the concept, let us consider the examples in the scenario of section 2. For instance, CEP can be used for simple events, like detecting the presence of a person in the house and triggering the discovery service to identify that person. However, whenever a person moves in the room, the presence event will be received. In order to prevent the triggering of the discovery service every time a person moves in the room, with CEP we can use windows of time. Using them, we can specify that we are only interested in those events every number of seconds or minutes.

Moreover, CEP can be used to find relationships between isolated events. For example, if the smoke detectors send an event and the temperature is above 40ºC, then we can assume that there is fire in the room and alert the user. Using again the windows of time, we could specify that if within 5 minutes the user has not responded to the alert, then it should trigger the sprinklers and send an alert to the fire department.

Finally, using CEP we can also configure some comfort rules according to user preferences. For example, if the user turns on the TV in a room with a window, and there is too much light outside, then it could close the blinds to improve the user's experience.

4. The DigiHome Service-Oriented Platform

The integration, mobility and adaptation issues impose several requirements for the development of smart homes environments. To deal with these issues, in this section we propose a comprehensive and simple solution called DigiHome, which enables the integration of events and context information as well as the dynamic configuration of applications. In particular, we propose a flexible architecture that modularizes the different concerns associated with event processing in ubiquitous environments by applying existing standards and approaches. In our solution, we support the integration of events sources (*e.g.*, sensors in our scenario), context providers (*e.g.*, mobile devices) and other kind of services (*e.g.*, actuators and reconfiguration services) implemented with a variety of technologies and interacting via different protocols by means of the SCA component model. Indeed, DigiHome deals with protocol heterogeneity, by enabling the incorporation at runtime of different communication mechanisms if required thanks to the SCA feature isolation of non-functional concerns.

In DigiHome, we follow the REST principles [3] to reduce the coupling between entities by focusing the interaction in the exchanged data, which can have multiple representations (*e.g.*, XML and JSON). In a similar way, for supporting the integration of devices with restricted capabilities, DigiHome promotes the usage of a lightweight API and simple communication protocols as stated by REST. In particular, our solution benefits from WSNs in order to process simple events and make local decisions when possible, by means of the Remora component model [9], which is a component model for WSNs based on SCA. Finally, the platform uses a CEP engine for the adaptation of applications and room configuration. Figure 2 depicts the general architecture of the platform. In the rest of the section we provide a detailed description of the different elements of the platform.

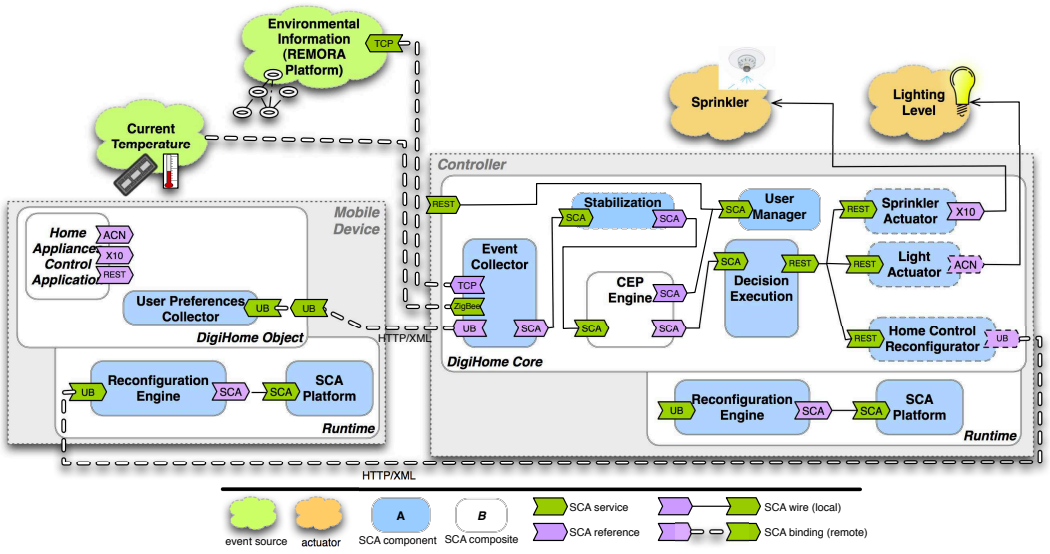

Figure 2. Description of the DIGIHOME architecture.

4.1. DigiHome Core

The core of the platform modularizes the main responsibilities for home monitoring. This means that the core contains the functionality required for event collecting, event processing, and deciding and executing the required adaptations of the applications deployed on DIGIHOME objects (cf. section 4.2) as well as the room configurations. In DigiHome, the Event Collector retrieves and stores the recent information produced by event and context sources, such as sensors and mobile devices. The CEP Engine is responsible for event processing and uses the Decision Executor to perform actions specified by the Adaptation Rules (defined in the CEP Engine). The CEP Engine also employes a User Manager service for determining if inhabitants have rights for executing specific system operations (*e.g.*, the activation of sprinklers or the alert of the fire department in situation 2) and who must to be notified in case of requiring human intervention. Furthermore the User Manager service is used by the Event Collector for deciding if the information from a mobile device has to be processed by the system or not.

On the other hand, the core contains different Actuator components that grant access to the available actuator services in the environment. Following a plug-in mechanism, these components can be installed or uninstalled at runtime. This means that the different actuators are optional, deployed according to the current service configuration and installed on different devices.

To enable the communication between different clients and to support the mobility of services and mobile devices, we incorporate ubiquitous bindings in SCA [10]. These bindings bring into SCA existing discovery protocols, such as UPnP [11] and SLP [12], providing the possibility to establish spontaneous communication. Furthermore, the ubiquitous bindings improve the context information advertisements with *Quality of Context* (QoC) [13] attributes for provider selection. Once the services are discovered, the ubiquitous bindings are flexible enough to allow the interaction via standard bindings, such as REST. The use of these ubiquitous bindings, as well as the modularization of the different concerns, makes it easy to distribute the different responsibilities in DigiHome.

4.2. DigiHome Objects

A DigiHome Object is an SCA component providing and/or consuming events to/from other DigiHome Objects. In our scenario, the mobile device executes a DigiHome Object that offers the user preferences as context information and hosts an adaptive application enabling the control of home appliances (that also consumes events indirectly in order to be adapted). The DigiHome Core can also be considered as a DigiHome Object. Because our solution is based on standards, and in hiding the service implementation with SCA, we can easily integrate other services in the smart home that are not part of the infrastructure (in particular, the actuators). In a similar way, we are exposing the DigiHome Objects via ubiquitous bindings so that other applications (that are not part of DigiHome) can benefit from the services offered by the platform.

The exchange of events between DigiHome Objects is done following a REST-based approach. This means that we exploit the simple REST interfaces (*i.e.*, PUT, POST, DELETE and GET) and unique identifiers (*i.e.*, URLs). In particular, the objects subscribe and unsubscribe via the POST and DELETE interfaces respectively. The subscription request includes the URL that is used for sending the events. The PUT operation is used in event notification. Because of the environment dynamism, the subscriptions have a configurable expiration time. If a subscription is not renewed, it will be discarded. Thus, this simple approach enables the usage of DigiHome Objects in different kinds of devices.

4.3. CEP Engine

To manage the events in our scenario, we need a decision-making engine that can process them and that can create relationships to identify special situations, using predefined rules. In order to identify the desired events, the CEP Engine requires to communicate with an Event Collector, which is in charge of dealing with the subscriptions to the event sources. If an adaptation situation is detected, a corresponding action is triggered, which can go from an instruction to an actuator, to the adaptation of the system by adding or removing functionality. These actions are received by the Decision Executor, which has the responsibility of communicating with the different actuators in the environment.

Because connections problems with the event sources are possible, the CEP Engine is configured with a set of rules and actions that are applied by default. For example, if the sensor movement detects the presence of someone in the room but the DigiHome core can

not detect her or his mobile phone, the system will apply a default rule for inhabitant presence and the associated actions according to the year season. These actions and rules can be modified at deployment and runtime (thanks to the FRASCATI reconfiguration capabilities).

In DIGIHOME, for the event processing in the Controller, we use ESPER [14], a Java open source stream event processing engine, to deal with the event management and decision making process. We chose ESPER for our platform because it is the most supported open source project for CEP and is very stable, efficient, and fairly easy to use. The following code excerpt shows an example of an ESPER rule used in our scenario, in section 2:

```
select sum(movement)
from MovementSensorEvent.win:time(60 sec)
```

This demonstrates the use of a time window, which is a moving interval of time. The rule collects all the events from the movement sensor from the last 60 seconds. By doing this, we can know if a user is still in the room or has already left, and adapt the room accordingly.

4.4. Support for Wireless Sensor Networks

In order to consume events from WSNs, we use the REMORA Component Framework [9]. This framework is an extension of SCA that brings component-based development into WSNs. The main motivation behind proposing REMORA is to facilitate *high-level and event-driven* programming in WSNs through a component-based abstraction. The primary feature of REMORA is provisioning a high-level abstraction allowing a wide range of embedded systems to exploit it at different software levels from operating systems to applications. REMORA achieves this goal by: *i)* deploying components within a lightweight framework executable on any system software written in the C language, and *ii)* reifying the concept of *event* as a first-class architectural element simplifying the development of event-oriented scenarios. REMORA meets efficiently the heterogeneity concerns related to WSN programming in the DIGIHOME platform as this model is SCA-compliant and portable to different operating systems used in the home sensor nodes.

REMORA proposes a TCP/IP mechanism to exchange events, which is encapsulated in an SCA component. We reuse this mechanism in order to define DIGIHOME objects for WSNs (so called Remora Objects), which are able to produce and consume simple events in the DIGIHOME platform. With these objects, we improve the efficiency of the system because the WSN is able to process simple events instead of going through the DigiHome Core for making local decisions (*e.g.* energy saving properties).

In particular, WSNs are equipped with a set of rules to monitor basic events, aggregate them, and emit the inferred events as global events to the CEP engine. The local rules are essentially elicited from the global rules identified based on the requirements of home owner, avoiding any potential conflicts between global and local rules. In addition to the use of local rules for node-level decisions, the core of our framework enables in-WSN decisions, whenever an event is required to be processed with other relevant events generated by other sensor nodes. For example, if a temperature sensor detects a high temperature, it needs to become aware of the smoke density in the room to know if there is a fire—*i.e.*, communicate with the smoke detecting

sensors. Furthermore, benefiting from the DigiHome modularization of concerns, as well as the transparent communication promoted by SCA, DigiHome objects can consume/notify events from/to Remora Objects with a small effort. Finally, Remora enables DigiHome to dynamically deploy and adapt the objects running on the WSN nodes at runtime.

5. Empirical Validation

Although the contribution of this article lies in the adoption of a versatile architecture style for integrating the diversity of device appliances available in the pervasive environments, we have also made a performance evaluation of a prototype, implementing the proposed platform. This experimentation demonstrates the reasonable overhead imposed by the DigiHome platform.

5.1. Implementation Details

We built a prototype of the DigiHome platform based on FraSCAti (cf. section 3.2). The selection of this platform is motivated by two main reasons: *i)* The platform brings reflection and reconfiguration capabilities at runtime into SOA systems and, *ii)* The FraSCAti customization capabilities according to the developer needs. The former is necessary in order to enable the dynamic adaptation of DigiHome applications. The latter allows us to easily have lightweight versions of DigiHome for executing them on devices with restricted capabilities, such as the mobile devices in the smart home scenario. In order to implement the ubiquitous bindings, we have used Cyberlink for Java* version 1.7 for UPnP and the jSLP library† for SLP. More detail about the ubiquitous bindings architecture and implementation can be consulted in [10]. Once the services are discovered, the DigiHome platform uses the data bindings for interacting with them. These data bindings follow a RESTful approach in order to exchange information [15].

5.2. Discovery and Communication Overhead

5.2.1. Test Bed Configuration

In order to test DigiHome, we have employed two MacBook Pro laptops, with the following software and hardware configuration: 2.4 GHz processor, 2 GB of RAM, AirPort Extreme card, Mac OS X 10.5.6 (kernel Darwin 9.6.0), Java Virtual Machine 1.6.0, and Julia 2.5.2. The mobile client used in the tests is a Nokia N800 Internet Table with 400 Mhz, 128 MB of RAM, interface WLAN 802.11 b/e/g, Linux Maemo (kernel 2.6.21), CACAOVM Java Virtual Machine 0.99.4, and Julia 2.5.2.

*Cyberlink for Java: http://cgupnpjava.sourceforge.net/
†jSLP: http://jslp.sourceforge.net/

5.2.2. Evaluation Results

We have implemented the situation 1 of the motivating scenario (cf. section 2) in order to measure the media latency for discovery (of preferences provider) and context dissemination in the DIGIHOME platform.

Table I reports these measures. We have executed 10000 successful tests, of which the first 100 were considered as part of the warm-up. In this setup, we retrieve the user preferences from multiple local and distributed providers and use multiple formats for the context information (*i.e.*, XML, JSON, and Java Object Serialization). In the local tests we executed the DigiHome core and the DigiHome objects in different virtual machines on the same laptop. In the distributed measures we used one laptop as Controller, and the other laptop and the Nokia device as information providers.

We also measured the delay for discovering the information provided by the sources. For discovery, we selected the UPnP and SLP protocols. In the tests, the platform aggregates the user's preferences to reduce the number of messages exchanged between the provider and the consumer. The measured time corresponds to the exchange of REST messages as well as the marshalling/unmarshalling of the information. The cost of executing others protocols, such as ACN and ZigBee was not considered in this article. The reader can find more information about the overhead introduced by these protocols in [16].

As seen, there is a linear increase of the latency with the different formats. This is a good characteristic of our solution, because we can integrate several entities with an acceptable overhead. We also observe that there is not a big variation in the communication cost between the different formats when the number of providers is low (until aprox. 10). As expected, when the providers are increased, the context exchange with object serialization is more efficient than the JSON and XML representations. Furthermore, the network usage introduces an overhead of approximately 300%.

Regarding the discovery cost (that includes the discovery time as well as the cost associated with the ubiquitous binding configuration), it is negligible compared to the context information retrieval, if there are not many providers. Our tests show that the use of SLP or UPnP for discovery does not have a big impact on the discovery time. In a similar way to the retrieval case, the measures including the network are bigger. Finally, we tested our solution using a Nokia Internet Table as a preference provider. As it can be seen, the use of this mobile device introduces an additional but still acceptable overhead for discovery and information exchange. This increase in cost is expected because of the limited resources of this kind of devices.

5.3. Event Processing Overhead

The latency for disseminating, as well as for discovering context, confirms that DIGIHOME can integrate heterogeneous entities with a reasonable performance overhead. Furthermore, according to the documentation provided by ESPER[14], the efficiency of the engine to process the events is such that it exceeds over 500,000 events per second on a workstation and between $70,000$ and $200,000$ events per second running on an average laptop. Thanks to the efficiency of the engine, the use of event processing in our system can be done at a low cost and given the modularity of our architecture, the ESPER engine can be installed in the device that provides

Table I. Performance of the DigiHome Platform.

Number of Providers	Retrieval Latency (Local Providers)			Retrieval Latency (Remote Providers)			Discovery Latency (Local Providers)		Discovery Latency (Remote Providers)	
	Objects (ms)	JSON (ms)	XML (ms)	Objects (ms)	JSON (ms)	XML (ms)	SLP (ms)	UPnP (ms)	SLP (ms)	UPnP (ms)
1	**38**	31	39	**101**	108	112	8	11	34	45
2	**46**	45	62	**160**	174	179	17	22	58	75
5	**65**	90	93	**361**	377	380	39	52	119	148
10	**145**	149	309	**418**	425	444	77	102	238	315
50	**504**	556	622	**1954**	1997	2039	361	481	1060	1410
100	**933**	1449	1500	**2269**	2286	2305	730	974	2194	2912
1 (N800)	N/A	N/A	N/A	**340**	376	N/A	N/A	N/A	129	136

the highest processing power. In the context of the DigiHome platform, we observed that Esper took $1ms$ on average to process the adaptation rules.

6. Related Work

6.1. Smart Home Solutions

Thanks to the increasing popularity of smart homes in the last years, we can find several solutions dealing with the integration of services in this kind of environments. For example, in [17] the authors propose ZUMA, a middleware solution providing universality, multi-user optimality, and adaptability. The authors claim that this solution based on clean abstractions for users, content and devices makes the integration of heterogeneous entities in smart homes easier. ZUMA defines a light-weight Device Control Protocol (DCP) that all the devices in the environment have to implement in order to be used by the middleware. The discovery is done by means of a registry that is part of the platform. In DigiHome, we do not impose a single protocol for interaction or discovery. Whereas, we based our solution on standard protocols and provide the flexibility to incorporate new protocols when required. These properties, combined with the advantages from SCA, foster the building of a more versatile solution for dealing with heterogeneity in smart environments.

Gaia [18] is a distributed middleware that provides similar functionality to an operating system. Gaia defines the concept of *active spaces* as geographic regions with limited and well-defined physical boundaries containing physical objects, heterogeneous networked devices, and users performing a range of activities. Examples of active spaces include meeting rooms and smart homes. The platform allows the coordination of software entities and heterogeneous devices in these active spaces. To do it, Gaia provides services for event management and distribution, context information query (for the context-based adaptation of applications), detection of digital and physical entities, storage of the information associate with entities, and file management. However, despite Gaia richness in terms of services, the solution is complex and the incorporation of new entities remains difficult. The simplicity of DigiHome

makes this task easy. Furthermore, GAIA lacks support for restricted devices, such as the mobile devices in our scenario, in contrast to DIGIHOME that uses SCA to do it. In [19], the MAVHOME (Managing an Intelligent Versatile Home) project is described. This project aims to build a home as a rational agent that maximizes inhabitants comfort and reduces operation costs. The project mainly focuses on prediction algorithms to guide decisions for controlling devices throughout the home. These algorithms include the *Smarthome Inhabitants Prediction* algorithm (that matches more recent sequences of events with stored sequences), *Active LeZi* algorithm (that applies information theory principles to process historical actions sequences) and a Task-Based Markov Model algorithm (for identifying high level tasks in action sequences). Benefiting from the DIGIHOME extensibility and concern isolation, these algorithms can be incorporated into the CEP engine in order to make our service-oriented platform more intelligent and autonomous in respect to the adaptation decisions.

6.2. Context Dissemination

In literature, it is possible to find two kinds of solutions that deal with context integration: *centralized* and *decentralized*. In the centralized category we can find middleware, such as PACE [20] and *Context Distribution and Reasoning* (ConDoR) [21]. PACE proposes a centralized context management system based on repositories. The context-aware clients can interact with the repositories using protocols, such as Java RMI or HTTP. For its part, ConDoR takes advantage of the object-oriented model and ontology-based models to deal with context distribution and context reasoning, respectively. In ConDoR, the integration of the information using different protocols is not considered as an important issue. The problem with this kind of approach is the introduction of a single point of failure into the architecture, which limits its applicability to ubiquitous computing environments.

On the other hand, in decentralized approaches we can find solutions like CORTEX [22] and MUSIC [23, 24]. CORTEX defines sentient objects as autonomous entities that have the capacity of retrieving, processing, and sharing context information using HTTP and SOAP. MUSIC middleware is another decentralized solution that proposes a peer-to-peer infrastructure dealing with context mediation. The decentralized approaches face the problem of fault tolerance by distributing the information across several machines. However, as well as some centralized solutions, the lack of flexibility in terms of the communication protocols remains a key limitation for these approaches. In addition to that, peer-to-peer approaches have performance and security problems. In DIGIHOME, we provide a solution, where the different interacting devices can process the events retrieved from the environment. Furthermore, in DIGIHOME we provide flexibility in terms of interaction by supporting different kinds of communication protocols and we also allow spontaneous interoperability.

6.3. Complex Event Processing

Given the increasing interest to integrate the flow of data into the existing systems, CEP has gained some attention as it can help to provide that integration transforming isolated data into valuable information. In this context we can find some works similar to ours in [25] and [26]. In [25], the authors integrate CEP into their existing project called SAPHE (*Smart and Aware*

Pervasive Healthcare), and also use ESPER as their CEP engine. As the project name shows, the project is applied to healthcare and uses sensors to monitor a patient's activity and vital signs. They use CEP to correlate and analyze the sensor data in order to calculate critical factors of the patient locally in their set-top box, without having to send all the events to an external server. In their approach they lack a way to discover new services and they never mention how, if possible, would they interact with actuators in order to adapt to the context and respond to a specific situation.

An *Event-Driven Architecture* (EDA) that combines the advantages of WSN with CEP is presented in [26]. They use an extension of the RFID EPCglobal architecture which allows the interaction of RFID and WSN events. Once the events are collected, they use CEP to detect specific situations. They use a smart shelf application as their scenario to show how the events from both sources can be combined. Even though both technologies seem to interact in their project, their specification is somehow limited because they do not specify how the information obtained could be used, other than generating a report that will be logged in the EPCIS server.

6.4. Wireless Sensor Networks

In [27], the authors describe a WSN-specialized resource discovery protocol, called DRD. In this approach, each node sends a binary XML description to another node that has the role of *Cluster Head* (CH). The CH is selected among all the nodes based on their remaining energy. Therefore, it is necessary to give all the nodes the capacity of being a CH. Consequently, all the nodes need an SQLlite database, libxml2 and a binary XML parser in order to implement the CH functionalities. In DIGIHOME, with our modular architecture, we consider the resource constraint of sensors nodes and provide a lightweight version of the platform based on the REMORA Framework that delegates complex processing to more powerful devices. Therefore, not all the nodes have to be CH. Furthermore, we benefit from the advertisement capacities of the sensor nodes to identify adaptation situations.

In CoBIs [28], business applications are able to access functionalities provided by the sensor nodes via web services. The major aim of the CoBIs middleware is to mediate service requests between the application layer and the device layer. The focus lies thereby on deployment and discovery of required services.

AGIMONE [29] is a middleware solution supporting the integration of WSNs and IP networks. It focuses on the distribution and coordination of WSN applications across WSN boundaries. AGIMONE integrates the AGILLA [30] and LIMONE [31] middleware platforms. AGIMONE is a general-purpose middleware with a uniform programming model for applications, that integrates multiple WSNs and the IP network. In our approach, we also promote the integration of sensor nodes via SCA bindings. Moreover, we enable spontaneous communications with some sensor nodes that execute a lightweight version of DIGIHOME.

7. Conclusions and Future Work

In this article, we have presented DigiHome, a platform addressing the mobility, heterogeneity, and adaptation of smart entities. In particular, DigiHome detects adaptation situations by integrating context information using an SCA-based architecture. This architecture promotes the modularization of concerns and fosters the application of the REST principles by exploiting the SCA extensibility. The simplicity and data orientation of REST, combined with the SCA independence of implementation technologies, make DigiHome an attractive solution to deal with heterogeneity in terms of interactions. The definition and application of ubiquitous bindings in the platform enable spontaneous communication by means of standard protocols (e.g.,UPnP and SLP), and furnish context provider selection (based on QoC attributes). On the other hand, the modularized architecture of DigiHome allows the definition of variants for the platform, called DigiHome objects, that can be deployed on resource-constrained devices. The functionality of these objects is exposed as services, accessible via several protocols, which can be accessed by clients that do not have to be part of the platform. Furthermore, the clear separation of concerns in the DigiHome architecture encourages the exploitation of WSNs for simple processing and local decision making. The suitability of our platform for context integration was evaluated with different discovery and context representations.

Future work includes further tests using some sensor nodes as context information providers, which will execute Remora objects objects. Also, we are currently working on a distributed CEP approach to diminish risk of failure, given the single instance of event processing in our project. We will also define more complex scenarios including user preferences conflicts and the absence of mobile devices when the presence of someone is detected in home. These scenarios will allow a further illustration of the potential of the platform. Furthermore, we plain to incorporate a replication mechanism of the DigiHome core by benefiting from the different devices such as desktops, laptops and set-top-box available in home. This mechanism will enable the platform to deal with possible failures of the device hosting the core. Finally, we plan to exploit the FraSCAti's reconfiguration capabilities in order to integrate new communication and discovery protocols at runtime.

Acknowledgement. This work is partly funded by the EGIDE Aurora and INRIA SeaS research initiatives.

REFERENCES

1. Romero D, Hermosillo G, Taherkordi A, Nzekwa R, Rouvoy R, Eliassen F. RESTful Integration of Heterogeneous Devices in Pervasive Environments. *Proceedings of the 10th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS'10)*, *LNCS*, vol. 6115, Springer, 2010; 1–14.
2. Open SOA. Service Component Architecture Specifications Nov 2007.
3. Fielding RT. Architectural Styles and the Design of Network-based Software Architectures. PhD Thesis, University of California, Irvine 2000.
4. Luckham DC. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Longman Publishing Co., Inc.: Boston, MA, USA, 2001.
5. Seinturier L, Merle P, Fournier D, Dolet N, Schiavoni V, Stefani JB. Reconfigurable sca applications with the frascati platform. *SCC'09: Proceedings of the IEEE International Conference on Services Computing*, IEEE Computer Society: Washington, DC, USA, 2009; 268–275, doi:10.1109/SCC.2009.27.

6. Mélisson R, Merle P, Romero D, Rouvoy R, Seinturier L. Reconfigurable run-time support for distributed service component architectures. *Automated Software Engineering, Tool Demonstration*, Antwerp Belgique, 2010. URL http://hal.inria.fr/inria-00499477/en/.
7. Bruneton E, Coupaye T, Leclercq M, Quéma V, Stefani JB. The FRACTAL Component Model and its Support in Java. *Soft. Pract. and Exp.* 2006; **36**(11-12):1257–1284.
8. OSGi Alliance. *Listeners Considered Harmful: The Whiteboard Pattern* Aug 2004.
9. Taherkordi A, Loiret F, Abdolrazaghi A, Rouvoy R, Le Trung Q, Eliassen F. Programming Sensor Networks Using REMORA Component Model. *Proceedings of the 6th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS'10) 6th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS'10)*, Santa Barbara, California, USA France, 2010; 15. URL http://hal.archives-ouvertes.fr/hal-00471516/PDF/Remora_DCOSS10.pdf.
10. Romero D, Rouvoy R, Seinturier L, Carton P. Service Discovery in Ubiquitous Feedback Control Loops. *Proceedings of the 10th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS'10)*, *LNCS*, vol. 6115, Springer, 2010; 113–126.
11. UPnP Forum. UPnP Device Architecture 1.0. http://www.upnp.org/resources/documents.asp Apr 2008.
12. Guttman E, Perkins C, Veizades J, Day M. Service Location Protocol, Version 2. RFC 2608 (Proposed Standard). http://tools.ietf.org/html/rfc2608 Jun 1999.
13. Krause M, Hochstatter I. Challenges in Modelling and Using Quality of Context (QoC). *Proceedings of the 2nd International Workshop on Mobility Aware Technologies and Applications*, Montreal, Canada, 2005; 324–333, doi:10.1007/11569510_31.
14. EsperTech. Esper. Http://esper.codehaus.org.
15. Romero D, Rouvoy R, Seinturier L, Loiret F. Integration of Heterogeneous Context Resources in Ubiquitous Environments. *Proceedings of the 36th EUROMICRO International Conference on Software Engineering and Advanced Applications (SEAA'10) 36th EUROMICRO International Conference on Software Engineering and Advanced Applications (SEAA'10)*, Michel Chaudron (ed.), ACM: Lille France, 2010; 4.
16. Zigbee Alliance. ZigBee and Wireless Radio Frequency Coexistence. http://www.zigbee.org/imwp/download.asp?ContentID=11745 Jun 2007.
17. Baker C, Markovsky Y, Greunen J, Rabaey J, Wawrzynek J, Wolisz A. Zuma: A platform for smart-home environmnents. *Intelligent Environments, 2006. IE 06. 2nd IET International Conference on*, vol. 1, 2006; 51 –60.
18. Román M, Hess C, Cerqueira R, Ranganathan A, Campbell RH, Nahrstedt K. A middleware infrastructure for active spaces. *IEEE Pervasive Computing* 2002; **1**(4):74–83, doi:http://dx.doi.org/10.1109/MPRV.2002.1158281.
19. Cook DJ, Youngblood M, Heierman EO III, Gopalratnam K, Rao S, Litvin A, Khawaja F. Mavhome: An agent-based smart home. *PERCOM '03: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*, IEEE Computer Society: Washington, DC, USA, 2003; 521.
20. Henricksen K, Indulska J, Mcfadden T. Middleware for Distributed Context-Aware Systems. *DOA'05: International Symposium on Distributed Objects and Applications*, Springer, 2005; 846–863.
21. Paganelli F, Bianchi G, Giuli D. A Context Model for Context-Aware System Design Towards the Ambient Intelligence Vision: Experiences in the eTourism Domain. *Universal Access in Ambient Intelligence Environments*, 2006; 173–191, doi:10.1007/978-3-540-71025-7_12.
22. Sorensen CF, Wu M, Sivaharan T, Blair GS, Okanda P, Friday A, Duran-Limon H. A context-aware middleware for applications in mobile Ad Hoc environments. *MPAC'04: Proceedings of the 2nd Workshop on Middleware for Pervasive and Ad-hoc Computing*, ACM: Toronto, Ontario, Canada, 2004; 107–110, doi:10.1145/1028509.1028510.
23. Hu X, Ding Y, Paspallis N, Bratskas P, Papadopoulos GA, Barone P, Mamelli A. A Peer-to-Peer based infrastructure for Context Distribution in Mobile and Ubiquitous Environments. *CAMS'07: Proceedings of 3rd International Workshop on Context-Aware Mobile Systems*, Vilamoura, Algarve, Portugal, 2007.
24. Kirsch-Pinheiro M, Vanrompay Y, Victor K, Berbers Y, Valla M, Frà C, Mamelli A, Barone P, Hu X, Devlic A, *et al.*. Context Grouping Mechanism for Context Distribution in Ubiquitous Environments. *DOA'08: Proceedings of the OTM International Conferences on Distributed Objects and Applications*, LNCS, Springer: Monterrey, Mexico, 2008; 571–588, doi:10.1007/978-3-540-88871-0_41.
25. Churcher GE, Foley J. Applying and extending sensor web enablement to a telecare sensor network architecture. *COMSWARE'09: Proceedings of the 4th International ICST Conference on COMmunication System softWAre and middlewaRE*, ACM: New York, NY, USA, 2009; 1–6, doi:10.1145/1621890.1621898.
26. Wang W, Sung J, Kim D. Complex event processing in epc sensor network middleware for both rfid and wsn. *ISORC'08: Proceedings of the 11th IEEE Symposium on Object Oriented Real-Time Distributed*

Computing, IEEE Computer Society: Washington, DC, USA, 2008; 165–169, doi:10.1109/ISORC.2008.59.
27. Tilak S, K Chiu NAG, Fountain T. Dynamic Resource Discovery for Wireless Sensor Networks 2005.
28. COBIS Consortium. Cobis. fp strep project ist 004270 2009. Http://www.cobis-online.de.
29. Hackmann G, Fok CL, Roman GC, Lu C. Agimone: Middleware support for seamless integration of sensor and ip networks. *DCOSS'06: International Conference on Distributed Computing in Sensor Systems*, Springer, 2006.
30. Fok L, Roman GC, Lu C. Mobile agent middleware for sensor networks: An application case study. *IPSN'05: Proceedings of the International Conference on Information Processing in Sensor Networks*, IEEE, 2006.
31. Fok CL, Roman GC, Hackmann G. A lightweight coordination middleware for mobile computing. *Coordination'04: Proceedings of the 6th International Conference on Coordination Models and Languages*, Springer, 2006; 135–151.